Remarks

Reconsideration of the application and allowance of all pending claims are respectfully requested. Claims 1-4, 9-11, 14-22, 27-30, 35-37, 40-48, 53-56, 58-61, 66-68 and 71-79 remain pending.

In the Office Action dated October 18, 2002, claims 1-4, 9-11, 14-22, 27-30, 35-37, 40-48, 53-56, 58-61, 66-68 and 71-79 were rejected under 35 U.S.C. 103(a) as being unpatentable over Schoening et al. (U.S. Patent No. 6,205,465; hereinafter, "Schoening") in view of Furlani et al. (U.S. Patent No. 5,995,998; hereinafter, "Furlani"). Applicants respectfully, but most strenuously, traverse this rejection for the reasons stated below and request reconsideration thereof.

An "obviousness" determination requires an evaluation of whether the prior art taken as a whole would suggest the claimed invention taken as a whole to one of ordinary skill in the art. In evaluating claimed subject matter as a whole, the Federal Circuit has expressly mandated that functional claim language be considered in evaluating a claim relative to the prior art. Applicants respectfully submit that the application of these standards to the independent claims of the present invention leads to the conclusion that the recited subject matter would not have been obvious to one of ordinary skill in the art based on the applied patents.

Applicants' recite a technique for managing thread pools of a computing environment (e.g., claim 1). This technique includes receiving from a first requester of the computing environment a request to be processed. This request is waiting on a response from a second requester of said computing environment, and the response is to be serviced by a thread pool selected from a set of one or more eligible thread pools. The technique further includes dynamically altering the set of one or more eligible thread pools to provide an altered thread pool set of one or more eligible thread pools, wherein a thread pool of the altered thread pool set is to service said response.

Applicants' invention thus includes, in part, one requester's request waiting on a response from another requester and dynamically altering a set of one or more eligible thread pools, thereby providing an altered thread pool set from which a thread pool is to service the response. Applicants respectfully submit that at least these features of the claimed invention are not taught, suggested or implied by Schoening or Furlani, alone or in combination.

Schoening describes a parallel processing technique for a multi-threaded environment in which threads are organized by receipt of execution components that have a partial order determined by preconditions and resource requirements. Further, the Schoening technique includes a partial order evaluator that resolves the partial order, thereby determining a final order of execution of these components, some of which may run in parallel on separate threads (see Abstract and col. 4, lines 1-29 thereof). Thus, the scheme in Schoening provides for executing components in parallel on separate threads and determining the order of execution of those components running in parallel. This is quite different from the present invention, which recites that a first requester's request to be processed is waiting on a response from a second requester. Schoening's request for initiating parallel processing of a component results in determination of whether that component can be run in parallel with another component, and placement on an appropriate thread, without waiting on a response. Since Schoening applies predefined rules to determine execution order at the startup of the network management system's ANI interface, this determination is performed prior to response processing, and thus is done without waiting on a response of any kind (col. 22, line 64 – col. 23, line 13). When the parallel processing mechanism of Schoening determines execution order, the determination is completed prior to the dispatch of threads (col. 41, lines 25-35), and thus, prior to any opportunity to wait on a response to be serviced by a thread (see also the steps preceding parallel execution in FIG. 5B, which fail to indicate that a request waits on a response and the related discussion at col. 42, lines 1-33). For the reasons stated above, applicants respectfully submit that Schoening does not teach, suggest or imply a first requester's request waiting on a response from a second requester.

Further, as noted above, Schoening fails to describe, suggest or imply dynamically altering the set of one or more eligible thread pools to provide an altered thread pool set of the one or more eligible thread pools, wherein a thread pool of the altered thread pool set is to

service the response from the second requester. The network management system in Schoening applies preconditions associated with execution components to determine whether the components can be run in parallel, and dispatches them on separate threads if parallel execution is determined (Abstract; col. 23, lines 14-21). Thus, Schoening describes assembling threads from execution components, but this is different from altering a set of one or more eligible thread pools, as recited in the present invention. Assembling threads in Schoening is directed to ordering the dispatch of components on threads, where some are run in parallel, without regard to whether the threads are selected from a set of thread pools that has been altered. In contrast, the alteration of a set of one or more eligible thread pools in the present invention is significant because a response to be serviced by a thread pool selected from the set prior to the alteration is to be serviced after the alteration by a thread pool of the altered set.

Not only does Schoening fail to disclose altering eligible thread pools, it also fails to teach, suggest or imply dynamically altering a set of one or more eligible thread pools for the purpose recited by applicants' invention. In Schoening, preconditions determine a "partial order" of execution components. This partial order is resolved into a final order of component execution by the partial order evaluator (see Abstract). Since the preconditions are defined and evaluated prior to run-time (col. 22, line 63 – col. 23, line 13), the application of such preconditions is not done "on the fly" and thus, is not a dynamic action. Further, once the final order of component execution in Schoening is determined and being serviced by one or more threads, those threads are not changed during run-time, and thus, are not dynamically altered (see, e.g., discussion of the parallel processing mechanism of FIG. 5A at col. 40, line 29 – col. 41, line 67 of Schoening). This is in contrast to the present invention, which recites dynamically altering the set of one or more eligible thread pools. Applicants submit that unlike Schoening, the alteration of the set of thread pools in the present invention occurs during run-time because the change in the source of the thread pool to service the response occurs while the request is waiting on the response from the second requester (i.e., while at least one thread is running).

In the Office Action, col. 35, lines 27-28 of Schoening is cited as teaching the request waiting on a response from a second requester. Applicants submit that this section discusses only the querying of a database, and includes no teaching or suggestion of waiting on a response

from a second requester. Relative to teaching altering eligible thread pools, the Office Action references col. 40, lines 43-45. Applicants respectfully submit that this section of Schoening discusses data sets over which execution can proceed independently, but includes no discussion of alteration of eligible thread pools. The Office Action also cites col. 42, lines 22-24; col. 41, lines 39-42; and col. 41, lines 3-14 as teaching altering thread pools. These sections of Schoening describe the workings of the parallel processing system, including an evaluation sequence (i.e., a partial order) that defines an order of execution of code blocks of a thread, and the selection of a timeBase that provides the partial order. These sections also describe the evaluation of the partial order, which results in a final execution order of certain functions, when parallel execution occurs. By defining the execution order of a thread, this discussion is directed to the assembly of a thread. As discussed above, the assembly of a thread is different from altering a set of eligible thread pools. Furthermore, none of these sections address dynamic alteration of one or more eligible thread pools, as claimed by the present invention. The Office Action also references applicants' Preliminary Amendment dated September 5, 2002 and cites col. 41, lines 11-12 of Schoening as describing eligible thread pools. Applicants respectfully submit that silence as to eligible thread pools alone is not essential to the remarks herein or in the Preliminary Amendment. Rather, as discussed above, applicants submit that Schoening fails to teach, suggest or imply dynamically altering a set of one or more eligible thread pools.

To summarize, Schoening does not teach or suggest a request from a first requester waiting on a response from a second requester, nor dynamically altering a set of one or more eligible thread pools. Thus, since Schoening fails to describe or suggest multiple aspects of applicants' claimed invention, Schoening does not render applicants' obvious. Further, applicants respectfully submit that Furlani does not overcome the deficiencies of Schoening as applied to the present invention.

Furlani describes a technique for locking interrelated objects in a multi-threaded computing environment. The technique monitors interrelationships between objects and provides a mechanism that locks the minimal set of objects needed for exclusive thread access. By minimizing the number of locked interrelated objects, unrelated objects are left unlocked and available to other threads, thereby minimizing lock contention (see Abstract and col. 2, lines 50-

65 thereof). This technique is very different from the one disclosed by applicants. For example, applicants respectfully submit that a careful reading of Furlani reveals no teaching, suggestion or implication of a first requester's request waiting on a response from a second requester, as claimed by applicants. Further, Furlani fails to describe or suggest dynamically altering a set of one or more eligible thread pools, as claimed by the present invention.

The Office Action alleges that Furlani teaches "[d]etermining different eligible thread groups or process configurations to complete a request" (paragraph 4, page 3, lines 10-11) and in the related discussion cites col. 6, lines 31-33 and col. 6, lines 44-45. These sections of Furlani describe the circumstances under which groups and group locks are merged. Applicants submit that the merging of groups and group locks in Furlani does not describe or suggest alteration of eligible thread pools, as claimed by applicants' invention. Groups in Furlani are very different from thread-pools claimed applicants. Furlani's groups are groups of objects (e.g., data structures) rather than pools of threads (i.e., independently executable parts of a program). Further, the merging of group locks in Furlani occurs, for example, when formerly independent objects A and B with separate locks become related. Once A and B become related, the separate locks merge so that a single lock applies to both A and B. This is very different from the present invention, as recited in claim 1, wherein the change in thread pools is not a merge, but a dynamic alteration of a set of one or more eligible thread pools into an altered thread pool set.

Based on the foregoing, since both Schoening and Furlani fail to teach or suggest multiple features of applicants' claimed invention, the combination of Schoening and Furlani also fails to teach or suggest multiple aspects of applicants' claimed invention.

For all of the above reasons, applicants respectfully submit that independent claim 1, as well as the other independent claims presented, are patentable over the combination of Schoening and Furlani. Additionally, the dependent claims are believed patentable for the same reasons as the independent claims from which they directly or ultimately depend, as well as for their own additional features.

For example, claim 19 further recites that the thread pool of the altered thread pool set is to service the response to avoid a deadlock with the request awaiting the response. An example of a deadlock situation is as follows: client A has a hold on Resource X (e.g., a lock on a file) and is currently updating Resource X. Client B then requests Resource X. The server breaks the lock by sending a callback to Client A. Client A responds to the callback before Client B is granted access to the resource. All threads in the thread pool may be already processing client requests that are waiting for the callback response from Client A. In this case, there are no threads in the thread pool to handle Client A's response, thereby causing a deadlock wherein Client A's response and the waiting requests are prevented from completing their respective processing (see specification, page 2, line 17 – page 3, line 4).

As recited by claim 19, deadlock is avoided when the thread pool of the altered thread pool set is to service the response, rather than the thread pool selected from the original set of one or more eligible thread pools (i.e., the set prior to the alteration). Thus, deadlock avoidance is an advantageous feature of dynamically altering the set of one or more eligible thread pools. In contrast, Schoening does not teach or suggest deadlock avoidance. Instead, Schoening is directed to a parallel processing technique that provides extensibility (i.e., supports new devices and new services without major revision of the network management system) and promotes efficiency (col. 3, lines 32-49). Although the Office Action cites col. 3, lines 24-30 as teaching the avoidance of deadlocks, applicants respectfully submit that this section is a statement of a problem without a corresponding solution included anywhere in Schoening. Applicants note that the only two other parts of Schoening mention deadlock (col. 54, lines 1 and 12-13). These two sections are limited to detection and notification of a deadlock situation without any discussion of avoiding deadlock, as claimed by the present invention. Applicants also submit that Furlani does not teach or suggest the avoidance of deadlock. Furlani's locking technique is directed instead to the minimization of lock contention, as discussed above. Thus, for the reasons stated above, applicants respectfully submit that the dependent claims presented herewith patentably distinguish over the applied art.

Should the Examiner wish to discuss this case with applicants' attorney, the Examiner is invited to telephone their below-listed representative.

Respectfully submitted,

Kevin P. Radigan
Attorney for Applicants
Registration No. 31,789

Dated: January 21, 2003

HESLIN ROTHENBERG FARLEY & MESITI P.C.
5 Columbia Circle
Albany, New York 12203
Telephone: (518) 452-5600
Facsimile: (518) 452-5579